What is claimed is:

1. A method for managing allocation of processors in a non-symmetric multiprocessor system, the multiprocessor system comprising a plurality of general-purpose processors and a plurality of special-purpose processors, each special-purpose processor having access to a local program store, the local program store being loaded with specific programs, the method comprising the steps of:

   a. compiling an application program in response to a request for execution of the application program, the application program comprising a plurality of interacting threads, each of the plurality of threads being capable of independently executing an application segment;

   b. scheduling the plurality of threads on various general-purpose processors and special-purpose processors based on the availability of the processors and the type of request; and

   c. managing the local program stores of each of the special-purpose processors for complying with processing load, the processing load being dependent on the requests for specific programs and the frequency of such requests.

2. The method as recited in claim 1 further comprising the step of forming a request-queue, the request queue storing all the stalled threads that have not been allocated a special-purpose processor, the stalled threads waiting for allocation of the special-purpose processors.

3. The method as recited in claim 2 wherein the step of scheduling the plurality of threads comprises the steps of:

   a. allocating a free general-purpose processor to a thread that does not request access to any special programs, the special programs being stored on local program stores of the special-purpose processors;

b. allocating a free special-purpose processor to a thread requesting access to a special program, the special program being stored in the local program store of the special-purpose processor being allocated; and

c. stalling the requesting thread and adding it to the tail of the request-queue, if no free processors are available.

4. The method as recited in claim 3 wherein the thread requesting access to a specific program loaded on a special-purpose processor is itself running on a general-purpose processor, the thread temporarily switching from the general-purpose processor mode to the special-purpose processor mode.

5. The method as recited in claim 3 wherein the step of allocating a free special-purpose processor comprises the steps of:

a. receiving an allocation request from a thread for a processor with a specific program loaded on its local program store;

b. searching for a free special-purpose processor with the requested program already loaded on its local program store;

c. allocating the free special-purpose processor with the requested program already loaded on its program store to the requesting thread, if such a processor is available; and

d. loading the requested program on the local program store of a free special-purpose processor and allocating it to the requesting thread, if no free special-purpose processor is available with the requested program already loaded on it.

6. The method as recited in claim 1 wherein the step of managing the local program stores comprises the steps of:

a. preferentially allocating a free special-purpose processor to a thread that requests access to a program already loaded on the local program store of the special-purpose processor; and

b. evicting the existing programs on the local program store of a free special-purpose processor until a space large enough to fit a specific program is created, in response to a request for a specific program not stored in the local program store of the special-purpose processor being allocated to the thread.

7. A method for allocating special-purpose processors in a multiprocessor computer system running an application, the application comprising a plurality of threads, each special-purpose processor having access to a local program store, the threads requesting access to special programs, the special programs having been stored on the local program stores of the special-purpose processors, the method comprising the steps of:

a. receiving an allocation request from a requesting thread for a special-purpose processor with a special program loaded on its local program store;

b. allocating a special-purpose processor with the requested program loaded on its local program store to the requesting thread, if a free special-purpose processor is available;

c. stalling the requesting thread and adding it to a request-queue, if no free special-purpose processors are available;

d. checking the request-queue for any pending requests, once a special-purpose processor is released by the requesting thread;

e. allocating the free special-purpose processor to the first thread in the request-queue that requests for a program already loaded on the processor;

f. allocating the free special-purpose processor to the first thread in the request-queue, if none of the threads in the request-queue request for a program already loaded on the processor; and

g. receiving the control of the allocated processor from the requesting thread, once the processor becomes idle.

8. The method as recited in claim 7 wherein the step of allocating a special-purpose processor with the requested program loaded on its local program store to the requesting thread, if a free special-purpose processor is available, comprises the steps of:

a. searching for a free special-purpose processor with the requested program already loaded on its local program store;

b. allocating the free special-purpose processor with the requested program already loaded on its local program store to the requesting thread, if such a processor is available; and

c. loading the requested program on the local program store of a free processor and allocating it to the requesting thread, if no free special-purpose processor is available with the requested program already loaded on its local program store.

9. The method as recited in claim 8 wherein the step of loading the requested program comprises the steps of:

a. virtually evicting the programs on the local program stores of all the free special-purpose processors until a processor with enough space on its local program store to fit the requested program is identified;

b. creating the space by actually evicting programs on the local program store of the identified special-purpose processor;

c. loading the requested program in the space created on the special-purpose processor; and

d. allocating the processor to the requesting thread.

10. The method as recited in claim 9 wherein the step of virtually evicting the programs from the local stores of free special-purpose processors is carried out in least-recently-used order, least-frequently-used order or first-in-first-out order.

11. The method as recited in claim 9 wherein the step of virtually evicting the programs from the local stores of free special-purpose processors further comprises the use of task information while creating space on the processor, the task information being information regarding task priority, task execution time, task pending time and program relevance.

12. The method as recited in claim 7 wherein the step of allocating the special-purpose processor to the first thread in the request-queue, if none of the threads in the request-queue request for a program that is already loaded on the local program store of the special-purpose processor, comprises the steps of:

a. virtually evicting the programs on the local program store of the special-purpose processor to create enough space for fitting in the requested program;

b. creating the space for fitting in the requested program on the processor by actually evicting the programs;

c. loading the requested program in the space created on the processor; and

d. allocating the processor to the requesting thread.

13. The method as recited in claim 12 wherein the step of virtually evicting the programs from the local store of the special-purpose processor is carried out in least-recently-used order, least-frequently-used order or first-in-first-out order.

14. The method as recited in claim 12 wherein the step of virtually evicting the programs from the local program store of the special-purpose processor further comprises the use of task information while creating space on the processor, the task information being information regarding task priority, task execution time, task pending time and program relevance.

15. The method as recited in claim 7 wherein one or more of the steps is embodied in a computer program product.

16. A system for managing allocation of processors in a non-symmetric multiprocessor environment, the multiprocessor comprising a plurality of general-purpose processors and a plurality of special-purpose processors, each special-purpose processor having access to a local program store, the system comprising:

a. a compilation service for compiling an application program in response to a request for execution of the application program, the application program comprising a plurality of interacting threads;

b. a processor allocation service for scheduling and synchronizing the plurality of threads on various general-purpose processors and special-purpose processors; and

c. a local program store managing service for managing the local program stores of each of the special-purpose processors for complying with processing load.

17. The system as recited in claim 16 wherein the processor allocation service comprises:

a. means for allocating a free general-purpose processor to a thread that does not request access to any special programs, the special programs being stored on the local program stores of special-purpose processors;

b. means for allocating a free special-purpose processor to a thread requesting access to a special program, the special program being stored on the local program store of the processor being allocated; and

c. means for stalling the requesting thread and adding it to the tail of the request-queue.

18. The system as recited in claim 16 wherein the local program store managing service comprises:

a. means for preferentially allocating a free special-purpose processor to a thread that requests access to a program already loaded on the local program store of the special-purpose processor; and

b. means for evicting the existing programs on the local program store of a free special-purpose processor until a space large enough to fit a specific program is created.